Patents, in a way, can seem like a gray area. For example, if an engineer designs a part for a product that a company needs to use, who actually gets the patent to that part? If the engineer was an employee of a company, say Whirlpool, and designed an advanced, efficient new part for the condenser of a refrigerator, you would think that the engineer could patent the product he designed. The engineer could sell his patented product to other appliance companies and make a lot of money. The patented product would technically be the engineer’s; however, he was assigned by Whirlpool to make the product. An engineer’s job is to design things and improve already existing things. Who really gets the ownership for the part?

This patent gray area gets even trickier when it comes to software. Open-source software that can be downloaded right from the Internet from a large company (Oracle and Java) to software a company may pay thousands for each year for employees (Microsoft Office). Who owns the software, and what can they do with it? If someone makes an app using Java, are they allowed to sell the app for their profit without owing money to Oracle since they own Java, and distribute it for free?

A new lawsuit came to the office of the Supreme Court in June 2014 regarding software patents. This case, *Alice V. CLS Bank*, basically stated “a computer-implemented method and system for facilitating settlements of financial transactions was unpatentable subject matter as an abstract idea”. The Supreme Court is basically saying that this situation with patenting software as it relates to financial transactions is a gray area, sort of like an abstract idea. After this lawsuit had occurred, patent infringement lawsuits dropped by 13% in 2014.

*Alice V. CLS Bank* happened relatively recently, almost 8 years ago in 2014. However, a similar case arose almost fifty years ago. This case was, *Gottschalk v. Benson*, was a lawsuit back in 1972 regarding software patents. The issue at hand was if a software innovation was created (application or other form of software), it should qualify to be patented by the creator. The issue comes down to machine code, if transforming from binary coded decimals to pure binary form was eligible for a patent. There were two claims to this issue: one claim was to carry out the code using the aid of a general use computer. The second claim was for that the method could be carried out using paper and pencil, without the use of a computer, would infringe it. The Supreme Court ultimately ruled that both of these claims failed to recite patentable matter.

The issue of software-related patents was revisited by the Supreme Court in 1981 in a case called *Diamond v. Diehr*. The case dealt with a patent regarding a rubber-curing process that involved the use of a software component. After this case, the Supreme Court basically ruled that anything created by humans, even through the use of a computer, could be patented as long as it had a tangible and useful result. From 1981 to the late 2000s, hundreds of thousands of software-related and computer-implemented patents were issued.

Since the Alice case, new guidelines have been added by the U.S. Patent and Trademark Office. It outlines more in-depth guidelines of what software-related inventions are patentable and those that are not. If Alice would have been a case in 1990 rather than 2014, many of the software-related patent claims filed with the U.S. Patent and Trademark Office would have been rejected considering how the court cases have changed what is patentable.

There are companies that have been attacked for their patents and have also defended the patents they own when contested by a rival company. For example, Costco defeated a patent from Mortgage Grader, Inc. The software at hand is a computer-implemented that shows users what their borrowing capabilities are for a mortgage. Capital One also successfully challenged a patent owned by Intellectual Ventures regarding a new system to help users budget.

Companies have also